//Search arr string

bool Search(string arr)

{

int node=0;

for(int i=0;i<arr.size();i++)

{

int id=arr[i]-'a';

if(tree[node].next[id]==-1)

return false;

node=tree[node].next[id];

}

return tree[node].endmark;

}

int main()

{

//ios\_base::sync\_with\_stdio(0);cin.tie(0);

#ifdef redback

freopen("C:\\Users\\Maruf\\Desktop\\

in.txt","r",stdin);

#endif

char arr[100];

ll t=1,tc;

read(tc); ///Test case

ll i,j,k,l,m,n;

while(tc--)

{

tree.pb(trie());

//Root node actually.

read(n);

//No of word in dictionary.

ll flag=1;

for(i=0; i<n; i++)

{

scanf("%s",&arr);

Insert(arr,0);

//Insert arr string into dictionary.

}

read(m); //No of queries.

for(i=0; i<m; i++)

{

scanf("%s",&arr);

if(Search(arr))

puts("Yes");

else

puts("No");

}

tree.clear();

//Delete The Memory.

}

return 0;

}

**Trie-Array:**

=======[ Theme ]=======

/\*

Dictionary contains N words.

Search if a string exists in Dictionary or Not.

2 Operation:

Insert: It inserts words in dictionary.

Search: It serches for words in dictionary.

\*/

struct trie

{

bool endmark;

int next[26];

trie()

{

endmark=0;

for(ll i=0; i<26; i++)

next[i]=-1;

}

};

node=tree[node].next[id];

}

tree[node].endmark=1; //endmark

marks the ending of a string

}

vector<trie>tree;

//Insert arr string

void Insert(string arr,int start)

{

ll node=0;

for(ll i=start; i<arr.size(); i++)

{

ll id=arr[i]-'a';

if(tree[node].next[id]==-1)

{

tree[node].next[id]=(int)tree.size();

tree.pb(trie());

}

node=tree[node].next[id];

}

tree[node].endmark=1; ///endmark marks the ending of a string

}

main()

{

#ifdef redback

freopen("C:\\Users\\Maruf\\Desktop\\in.txt","r",stdin);

#endif

ll t=1,tc;

read(tc);

ll i,j,k,l,m,n;

while(tc--)

{

scanf("%s",&s);

scanf("%s",&p);

ll ans=KMP();

printf("Case %lld: %lld\n",t++,ans);

}

return 0;

}

/\*

Input:

4

axbyczd

abc

abcabcabcabc

abc

aabacbaabbaaz

aab

aaaaaa

aa

Output:

Case 1: 0

Case 2: 4

Case 3: 2

Case 4: 5

\*/

**Trie output:**

Yes

No

No

Yes

Yes

No

**Trie input:**

1

3

aaa

asdasd

sssssss

6

aaa

aa

aaaa

asdasd

sssssss

ssasd

**KMP:**

ll fail[NN]; //Highest Prefix which equals as postfix at i.

char s[NN]; //Main String.

char p[NN]; //Find P-string in main string.

void failure(void)

{

ll i,j,k,l;

ll m=strlen(p);

fail[0]=0;

ll q=0;

for(i=1;i<m;i++) //starts from 1, not 0.

{

while(q>0 && p[i]!=p[q])

q=fail[q-1];

if(p[i]==p[q])

q++;

fail[i]=q;

}

return;

}

ll KMP(void)

{

ll i,j,k,l,cnt=0;

ll m=strlen(p);

l=strlen(s);

failure();

ll q=0;

for(i=0;i<l;i++)

{

while(q>0 && s[i]!=p[q])

q=fail[q-1];

if(s[i]==p[q])

q++;

if(q==m)

{

q=fail[q-1];

cnt++; //we got one substring

}

}

return cnt;

}

**Tenplate New:**

/\*\*

\* @author : Maruf Tuhin

\* @College : CUET CSE 11

\* @Topcoder : the\_redback

\* @CodeForces : the\_redback

\* @UVA : the\_redback

\* @link: http://www.fb.com/maruf.2hin

\*/

#include <bits/stdc++.h>

using namespace std;

typedef long long ll;

typedef unsigned long long llu;

#define ft first

#define sd second

#define mp make\_pair

#define pb(x) push\_back(x)

#define all(x) x.begin(),x.end()

#define allr(x) x.rbegin(),x.rend()

#define mem(a,b) memset(a,b,sizeof(a))

#define inf 1e9

#define eps 1e-9

#define mod 1000000007

#define NN 100010

#define read(a) scanf("%lld",&a)

int main()

{

//ios\_base::sync\_with\_stdio(0); cin.tie(0);

#ifdef redback

freopen("C:\\Users\\Maruf\\Desktop\\in.txt","r",stdin);

#endif

ll t=1,tc;

//read(tc);

ll i,j,k,l,m,n;

while(tc--)

{

}

return 0;

}

# String Multiplication (Dipto):

string subtract(string a,string b)

{

int cmp=compare(a,b);

if(cmp==0) return "0";

if(cmp==-1) swap(a,b);

string ans;

int tmp,borrow=0;

for(int i=a.size()-1,

j=b.size()-1; i>=0||j>=0; i--,j--)

{

if(j<0) tmp=a[i]-'0';

else tmp=a[i]-b[j];

if(borrow)

{

tmp--;

borrow=0;

}

if(tmp<0)

{

tmp+=10;

borrow=1;

}

ans+=tmp+'0';

}

reverse(ans.begin(),ans.end());

ans=remove\_leading\_zeros(ans);

if(cmp==-1) ans="-"+ans;

return ans;

}

# String Addition (Dipto):

string add(string a,string b)

{

string ans;

int tmp,carry=0;

for(int i=a.size()-1,

j=b.size()-1; i>=0||j>=0; i--,j--)

{

if(i<0) tmp=carry+b[j]-'0';

else if(j<0)

tmp=carry+a[i]-'0';

else

tmp=carry+(a[i]-'0')+(b[j]-'0');

carry=tmp/10;

ans+=(tmp%10)+'0';

}

if(carry) ans+='1';

reverse(ans.begin(),ans.end());

return ans;

}

# Substraction:

int compare(string a,string b)

{

if(a.size()>b.size()) return 1;

if(a.size()<b.size()) return -1;

for(int i=0; i<=a.size()-1; i++)

{

if(a[i]>b[i]) return 1;

if(a[i]<b[i]) return -1;

}

return 0;

}

string remove\_leading\_zeros(string str)

{

int zero=0;

for(int i=0; i<str.size(); i++)

{

if(str[i]=='0') zero++;

else break;

}

if(zero==str.size()) return "0";

return str.substr(zero);

}

string subtract(string a,string b)

{

int cmp=compare(a,b);

if(cmp==0) return "0";

if(cmp==-1) swap(a,b);

string ans;

int tmp,borrow=0;

for(int i=a.size()-1,j=b.size()-1; i>=0||j>=0; i--,j--)

{

if(j<0) tmp=a[i]-'0';

else tmp=a[i]-b[j];

if(borrow)

{

tmp--;

borrow=0;

}

if(tmp<0)

{

tmp+=10;

borrow=1;

}

ans+=tmp+'0';

}

reverse(ans.begin(),ans.end());

ans=remove\_leading\_zeros(ans);

if(cmp==-1) ans="-"+ans;

return ans;

}